Tâche2 : Tous les mesures ont été implémenter par nous-même. Durant la prise de mesure, le code prend comme entrain le dossier du code (et non des tests) et évalue chacun des fichier java à l’intérieur. Tous les fichiers non-java ne sont pas pris en compte.

Q3 : Pour évaluer la maturité du code nous avons utiliser comme métriques le nombre de jours entre aujourd’hui et des vingt derniers commits sur le code et le nombre de lignes ajoutées et supprimées de ceux-ci. Si le nombre de jours moyen est très grand, cela nous indique que le projet doit être mature car il n’y pas beaucoup de travaille à faire. Le nombre de lignes modifiées vient confirmer le tous. Lorsque ce nombre est grand, cela veut dire que le code est en construction. Peu de ligne de codes indique souvent que seulement des bogues ont été réglés, synonymes souvent de support pour un projet mature.

Q3 : Nous avons comme résultats une médiane de 286 jours entre le moment ou nous avons rouler le code et le moment des commits. Nous semblons avoir un code mature car il y a eu peu de code fait durant les derniers jours. De plus, le nombre médian de ligne supprimé est 12.5 et le nombre de lignes ajouté est de 6.5, donc la plus parts de ces commits était très petit, indiquant une fois de plus un signe de maturité dans le projet.

Q4: Le code semble pourvoir être en grande partie automatiquement tester. Nous avons 31% pour le PMNT si nous excluons les interfaces, les classes privées et les autres types de fichier java qui ne sont pas des classes. Cette valeur n’est pas catastrophique mais plus proche de 0 aurait été mieux. Pour accompagner cette métrique, nous avions regarder le ratio de taille de code sur taille de test. Elle est un proxy qui nous indiquer si les tests englobent beaucoup de cas. Nous avons comme valeur 2.8 qui semble indiquer que les tests sont accès exhaustifs